Ann. of Appl. Math.
36:1(2020), 31-47

AN EFFECTIVE DETAILED ROUTING ALGORITHM CONSIDERING ADVANCED TECHNOLOGY NODES*

Xiqiong Bai[1], Dixiu Xiao[1], Jianli Chen[2], Wenxing Zhu[2]†, Yadong Zhang[3], Taotao Lu[3], Lifeng Wu[3]

(1. *College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350116, Fujian, PR China;*

2. *Center for Discrete Mathematics and Theoretical Computer Science, Fuzhou University, Fuzhou 350116, Fujian, PR China;*

3. *Empyrean Software, Inc., Beijing 100000, PR China*)

Abstract

Detailed routing has become much challenging in modern circuit designs due to the extreme scaling of chip size and the complicated design rules. In this paper, we give an effective algorithm for detailed routing considering advanced technology nodes. First, we present a valid pin-access candidates generation technology for handling complex pin shapes. Then, we propose a tree-based nets components selection algorithm to decide connecting order for multiple nets components. Finally, combined with global routing results and advanced technology nodes, an initial routing results optimization algorithm is presented to achieve the final detailed routing results. Experimental results on industry benchmarks show that, our proposed algorithm not only achieves 100% routability on real industrial cases in a reasonable runtime, but also optimizes total wirelength, total vias and other advanced technology nodes simultaneously.

Keywords detailed routing; advanced technology nodes; pin-access; total vias

2000 Mathematics Subject Classification 68Q25

1 Introduction

Routing is considered as the most time-consuming and important stage in the VLSI design flow. In addition, with ever increasing requirements, many new design rules are introduced to satisfy modern industrial demands. Due to the complexity

*Manuscript received September 29, 2019
†Corresponding author. E-mail: wxzhu@fzu.edu.cn

of the routing problem, routing is usually divided into two stages: global routing and detailed routing. During the global routing stage, nets are routed on a coarse-grain grid structure with the objective of determining the regions within which each net will be routed. After an approximate routing solution is determined for each net, the detailed routing stage is to find the exact routes of all nets [1]. Since detailed routing is generated based on the global routes, the quality of the final interconnects depends largely on the quality of the global routing solution [27].

Considering advanced technology nodes in detailed routing is a complicated step in the physical design process. A high-performance chip requires that several corresponding metrics need to be evaluated and considered in this dead-or-alive process. With the aim to achieve a better detailed routing result, honoring global routing results can maximize reducing the disturbance to these metrics (e.g., timing, routability [3], manufacturability, skew, and congestion [2, 15, 16]), and so on. Figure 1 is a comparison of routing results of whether or not to take into account the impact of congestion for a net with four pins. If the detailed router routes wires over the region as shown in Figure 1 (a), it will have overlapped wires because of congestion. Figure 1 (b) is a modified detailed routing result considering advanced technology nodes. With the rapid development of modern industrial tools and lithography requirements, satisfying all advanced technology nodes in a detailed routing process is becoming more and more challenging. Therefore, several corresponding metrics need to be managed in different steps during the detailed routing process to meet all constraints in the final detailed routing result.

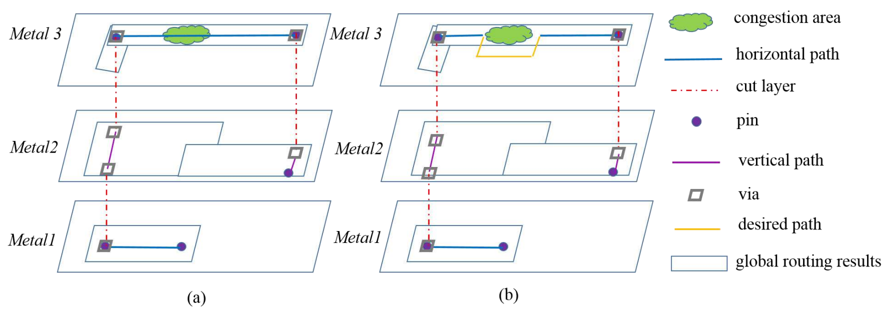


Figure 1: (a) A detailed routing result without considering congestion.
(b) A modified detailed routing result by considering congestion area.

1.1 Previous work

Many works have been presented for VLSI routing based on the shortest path algorithms, which can be divided into two categories: maze routing algorithm and line-search algorithm [4, 5]. The fundamental maze algorithm is *Lee's* algorithm [6], which is an application of the breadth-first search method. This algorithm consists of two main phases, and is applied to a uniform grid with known starting and ending points of a net. In the first phase, adjacent points of the starting point of the net are marked by label 1. Then after each step i, the unmarked neighbors of the labeled elements are labelled by $i + 1$. This phase will be repeated until the target point is found. The shortest path is produced by backtracking diminishing labels from the target point to the beginning point in the second phase. If there exist several neighboring points with the same label, the guidance for choosing the next point is to follow the path's direction and decrease the number of bends at the same time.

After that, by applying the A* heuristic search, Hadlock [22] proposed a shortest path algorithm called the Minimum Detour (MD) algorithm, which is guided by the number of detours during the path search process. Compared to $Lee's$ algorithm, the MD algorithm not only costs less time and the number of detours during the shortest path stage, but also achieves a better result. Different from the MD algorithm, Soukup's algorithm [8] combined advantages of the depth-first search and the breadth-first search, which improves the runtime of the original maze-routing algorithm. However, since it is a heuristic search method, the path found is usually not the shortest.

The search lengths at each step of all the above maze-routing algorithms are unit grid length, which adds the burden of storage and search ability during the routing process. As a result, line-search algorithms have been proposed to change this situation. The search segments are series of effective lines rather than unit grid intervals. Hence they can save runtime and memory for finding a single-shaped path, but usually cannot ensure to find a shortest path. For example, $Hetzal's$ algorithm [9] is an improved A* algorithm which searches intervals instead of nodes. Intervals are generated by clustering one row or one column of consecutive identical cost nodes, and are stored before the path searching process. However, $Hetzal's$ algorithm was proved that it does not enhance the running time [10]. What's more, these algorithms are mainly used for graph grids and the corresponding solution may be limited. So they cannot meet our research requirements for detailed routing with advanced technology nodes.

In addition, the above mentioned algorithms are mainly the path search process between two points. However, routing terminals of a net may not be a single source and a target, and the characteristics of multi-terminal nets indicate that our routing problem is an NP-complete problem [11]. Hence, basic methods of the Steiner tree problem have been widely promoted to minimize the total wirelength and consider other constraints. Typical approach to route a multi-terminal net is usually divided into two steps, that is, build a Steiner tree firstly, and then search a path between two points based on all pairs of connections of the tree.

Many routing methods have been proposed to solve the multi-terminal net routing problem, which includes two main types of algorithms: sequential and concurrent routing algorithms. Sequential routing algorithm has the advantage of simpleness, but routing one net may affect routability of other nets or degrade routing quality of all nets. To avoid this issue, another kind of algorithms attempt to find the final results of all nets concurrently. Shragowitz [12] proposed an algorithm based on the multicommodity flow formulation for two-terminal nets. Raghavan [13] presented an improved algorithm which can handle three-terminal nets. Later, Albrecht [14] gave a rigorous algorithm for global routing based on a new approximation algorithm for the multi-commodity flow problem.

In VLSI detailed routing, the solution space is much smaller than that of global routing. Since it is rather hard for an optimization method to consider global routing and advanced technology nodes simultaneously, it is great to design efficient detailed routing algorithms for producing high quality routing solution with considering advanced technology nodes.

1.2 Our work

In this paper, we design an effective algorithm flow to achieve a detailed routing result with global routing information and advanced technology nodes. Based on the circuit netlist and global routing results information, our algorithm connects all nets efficiently and optimizes the total wirelength, the total number of vias, the out-of-rectangle vias and wirelength, wrong wires and off-track vias. In our algorithm, we identify shape-based pin-access candidates by considering spacing violation and remaining enough available routing space. After preprocessing of all valid points for nets, tree-shape structure is used to select sequence of connections for optimized nets with the aim to decrease our routing area and wirelength. Finally, we propose an optimized detailed routing process by considering global routing information and advanced technology nodes. The main contributions of our work are summarized as follows:

- A valid pin-access candidates generation algorithm is proposed to obtain valid pin-access candidates from different shape of pins within track grids. Since some connection violations may have a large effect on later detailed routing stage, after generating pin-access candidates, spacing violation-aware candidates restriction is applied to achieve valid pin-access candidates.

- A tree-based nets components selection algorithm is applied to select net's components points by reducing total wirelength and total vias. We first divide nets multiple components into several pairs with corresponding weights, then establish connection points of these components based on known division. This algorithm can reduce abstract distance between two points for each pair of components, and decrease the total wirelength and total vias for the whole net.

- An algorithm is designed for optimizing initial routing results. Initial detailed routing solution is produced by carrying out a rectangle-driven A* algorithm at first, then based on via candidates, detailed routing optimization is presented to honor global routing results and consider advanced technology nodes for the whole net, with the aim to reduce out-of-rectangle wirelength, number of vias and other constraints.

- Experimental results show that our algorithm can achieve 100% routability in a reasonable runtime, while handling several advanced technology nodes with optimized total wirelength and total vias on industrial benchmarks.

The rest of this paper is organized as follows. Section 2 is the preliminaries. Section 3 describes our algorithm in detail for our complicated detailed routing problem. Empirical studies of our proposed algorithm are presented in Section 4. Finally, conclusion is given in Section 5.

2 Preliminaries

2.1 Modern connection constraints

In this subsection, several modern connection constraints and routing preference metrics will be described for satisfying the industrial advanced technology nodes on global routing results.

1) **Open Net.** The pins of each net need to be fully connected. If any pin in a net is disconnected, the net will be considered as an open net, and our detailed router will regard this kind of nets as failed wires.

2) **Cut Spacing.** The cut spacing specifies the minimum spacing distance between every two vias on cut layers, which ensures that vias on the same net or different nets must be placed satisfying the minimum spacing distance.

3) **Min Area.** By specifying the min area rule for all routing segments, the area of each routing polygon must be equal to or bigger than the default minimum value. A *patch* may be added to the polygon in order to increase its area up to the minimum default value.

4) **Short Area.** In our industrial benchmarks, each metal layer has only one value of width. Namely, a short area violation will happen when either a via or metal wire overlaps with another via model, metal wire, blockages, or pin shapes.

These cases can change the stable width value, and all intersection part of these cases are the short area.

5) **Rectangle Spacing.** All rectangles on routing layers have a default minimum spacing value between every two objects, such as the spacing between two routing segments, the spacing between wires and obstacles, the spacing between vias and obstacles, and so on.

An example of detailed routing result which violates connection constraints is presented in Figure 2. These situations indicate that every routing segment needs to satisfy specific connection constraints by considering advanced technology nodes during the detailed routing process.

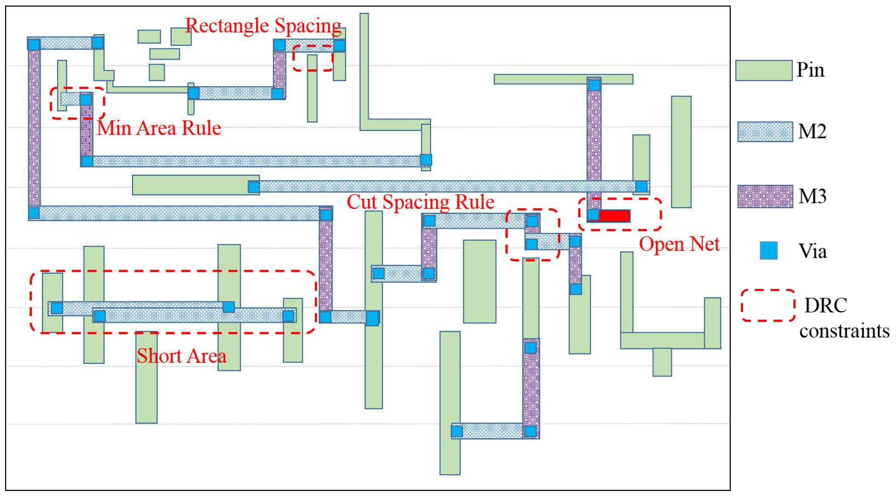


Figure 2: A detailed routing result with several connection constraints

2.2 Routing preference metrics

Due to modern connection constraints and other limited routing resources, we use several routing preference metrics [17] to evaluate the detailed routing quality, such as out-of-rectangle wirelength and vias, wrong-way routing and off-track routing. The distinction between routing preference metrics and modern connection constraints is the influence on routing results. By optimizing these metrics during the detailed routing process we can improve the quality of detailed routing results. We describe these metrics in detail in the following concepts.

1) **Out-of-Rectangle Wirelength and Vias.** Following global routing results is a wise choice for detailed routing process. Since global routing results in our industrial benchmarks are lists of rectangles which are saved in a specific file, each list of rectangles of one net can ensure to cover a detailed routing solution of the associ-

ated net. However, due to the congestion location of pins and limited global routing space, out-of-rectangle wires and vias are acceptable to accomplish all routing nets while considering other connection constraints.

2) **Wrong-Way Routing.** All routing layers have a preferred routing direction, which is either horizontal or vertical to guide routing path on the corresponding metal layer. What's more, routing directions on adjacent metal layers must be mutually perpendicular to each other to satisfy modern chip structure. If a horizontal wire exists on a vertical metal layer, this horizontal wire is called a wrong-way wire. Wrong-way wires may lead to a bad final routing result.

3) **Off-Track Routing.** Each metal layer has its track grids for assigning routing resources within a given wiring range. This means off-track routing wires are not aligned with track grids, and vias located outside the given track grids are off-track vias. Since the via structure is made up of two adjacent metal layers and a cut layer, routing wires on different layers needs to abide by specific tracks grids of different layers at the same time.

2.3 Problem statement

Given a set of m nets $N = \{n_1, n_2, \cdots, n_m\}$, each net has its specific pin information, and a set of m global routing results $G = \{g_1, g_2, \cdots, g_m\}$, which are comprised by several rectangles. The goal of our detailed routing problem is to achieve a routing result for each net $n_i \in N$ while considering its corresponding global routing result $g_i \in G$, and each net also should optimize the following metrics simultaneously: (1) all nets need to guarantee no open nets or short areas in our detailed routing results; (2) the total wirelength of all nets; (3) the total number of vias; (4) the out-of-rectangle vias and wirelength; (5) wrong wires and off-track vias.

In this paper, our detailed routing problem is based on the 3-D grid graph structure with unit length of grid per layer. Assume that the preferred direction of metal 1 is horizontal, and adjacent metal directions are perpendicular to each other. Our routing capacity of the unit track grids is only those unblocked edges with corresponding routing preference direction.

All benchmarks are derived from a single-core 32-bit processor with four memory cores, and multi-terminal nets are included without power and ground (PG) nets. Complex pin shapes exist not only in a single metal layer, but also in multiple metal layers. What's more, the number of blockages and layers, the die area of routing have increased the difficulty of our routing problem. The complicated industrial benchmarks indicate that line-search algorithms may not be applied to our problem, since they have designated routing direction and cannot route all nets successfully.

3 Our Algorithm

The overall flow of our algorithm is summarized in Figure 3. It starts with preprocessing circuit netlist and global routing results information on track grids. Circuit netlist is made up of specific pin information of all nets, and every pin information is decided by its relative orientation and location on corresponding boxes. A list of rectangles with assigned layers represents the regions passed by the global routing result of the associated net, which guarantees to cover at least a fully connected detailed routing solution for the net. After completing the above processes, our routing resources are divided into several kinds of points and our algorithm consists of three main parts: (1) valid pin-access candidates generation, (2) tree-based nets components points selection, (3) optimizing initial routing results. We will detail these three major parts in the following subsections.

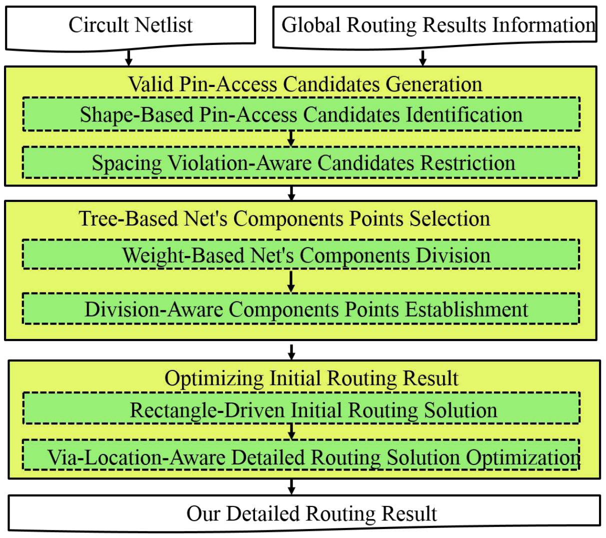

Circuit Netlist
Global Routing Results Information
Valid Pin-Access Candidates Generation
Shape-Based Pin-Access Candidates Identification
Spacing Violation-Aware Candidates Restriction
Tree-Based Net's Components Points Selection
Weight-Based Net's Components Division
Division-Aware Components Points Establishment
Optimizing Initial Routing Result
Rectangle-Driven Initial Routing Solution
Via-Location-Aware Detailed Routing Solution Optimization
Our Detailed Routing Result

Figure 3: Algorithm flow

3.1 Valid pin-access candidates generation

In this subsection, to reduce the complexity of this process, we define the concepts of pin-access candidate and valid pin-access candidate [18,19] as follows.

1) **Pin-Access Candidate:** A pin-access candidate on a metal layer is the intersection point of every two adjacent track grids within corresponding pin shapes. Pin-access candidates on the same single layer are put into a vertex set, and through this step, multiple sets of vertices corresponding to different layers are generated for the entire chip. In our pin-access candidates generation step, two pin shapes never share the same pin-access candidate, but one pin shape may have many pin-access candidates.

2) **Valid Pin-Access Candidate:** A pin-access candidate is valid if we can place at least one type of vias at that point without violating some connection constraints, such as spacing distance between vias and other objects. Otherwise, these pin-access candidates are considered invalid.

We intend to achieve valid pin-access candidates P_{vc} for each net with their global routing information and spacing constraints. Since the off-track wires and wrong wires may have a negative effect on routing result, we firstly present a technology to build track grid with routing direction for each layer. Specifically, a track grid T_g for a metal layer is created uniformly which covers the entire design area from bottom to top (or left to right). Secondly, all pin access candidates P_c are created by the intersection of points in pin shapes of every two adjacent track grids.

After creating pin-access candidates by track grids and pin information, we use Algorithm 1 to select valid pin-access candidates. Based on the generated vertex sets of pin information P_i and preprocessed track grids T_g, we generate pin-access candidates for every pin at the beginning, then inspect set of vertexes P_{ci} with spacing constraints. All candidates violating spacing constraints are saved in P_{vs} for later judgment. If no points are created within track grids, then a unit track interval is added to expand our search scope each time. We take T_u in pseudo code to count the number of expansions. After shape-based pin-access candidates are created, these candidates P_{vs} violating spacing constraints are removed.

```
Algorithm 1: Valid Pin-Access Points
Require: $P_i, T_g, P_{vs}$
Ensure: $P_{vc}$
 1: $T_u \leftarrow 0, p_{vc(used)} \leftarrow 0, P_{vc} \leftarrow 0$;
 2: while $P_i$ is not empty
 3:   for every pin in $P_i$
 4:     $P_c \leftarrow$ Get pin access candidates;
 5:     for every $P_{ci} \in P_c$
 6:       if $P_{ci} \in P_{vc(used)}$ or $P_{ci} \in P_{vs}$
 7:         $P_{vc} \leftarrow P_c \setminus P_{ci}$;
 8:         $P_{vc(used)} \leftarrow P_{vc(used)} \cup P_{ci}$;
 9:         $T_u \leftarrow T_u + 1$;
10:       end if
11:     end for
12:   end for
13: end while
14: return $P_{vc}$
```

Figure 4 is an example to find valid pin-access candidates considering specific spacing rules. Since two generated pin-access candidates in Figure 4 (a) cannot satisfy the spacing constraint, a unit track interval is expanded in Figure 4 (b) on the second point to produce our valid pin-access candidates, in which the purple point is our valid pin access point for this pin.

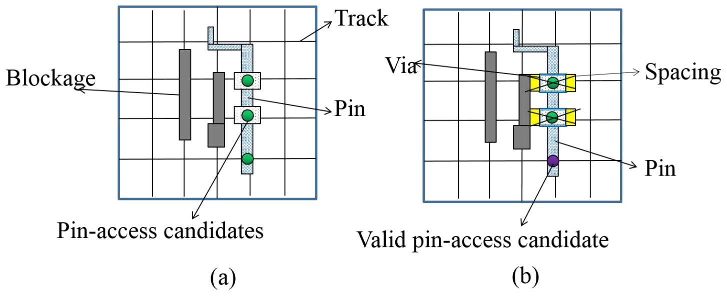


Figure 4: An example to find valid pin-access points

In addition, if the density of obstacles on the first metal layer is high, we may prefer to route less nets on this layer for routability of later process. Thus, if valid pin-access candidates are located in the range of global routing results, a via can be directly generated to start routing on the upper metal. Otherwise, we need to choose valid pin-access candidates closest to the global routing results and to find the shortest path to connect them on the first metal layer.

3.2 Tree-based net's components points selection

Components of a net are pins of this net. After the first step for finding valid pin-access candidates, the second step is to select a point of each pin for each net by optimizing total wirelength L_{total} and the total number of vias V_{total}.

3.2.1 Net's components division

With a set of n pins on a net, a graph $G(V, E)$ is constructed to divide each net into tree shape. Vertexes of the graph are composed of net components, and every two vertexes has an edge between them. Owing to the complexity of pin shapes, we set coordinates of all vertexes as the center points of the rectangles. With the aim to reduce total wirelength and total vias, connecting two far vertexes and searching random directions are meaningless.

Our connecting order of these vertexes starts from choosing the leftmost candidate of all nets, then Prim algorithm is applied to construct the minimum spanning tree (MST) on the above graph for each net, which aims to search for a minimum spanning tree within the weighted connected graph. The tree formed by the edge subset includes not only all the vertices in the connected graph, but also the sum of the weights of all the edges is the smallest.

In our algorithm, the weight for each edge is the Manhattan distance between two vertexes and the via number calculated from the global routing results. We define the weight of a via as 5, and the weight of Manhattan distance of an edge between two points is 1. Then, the edge weight is

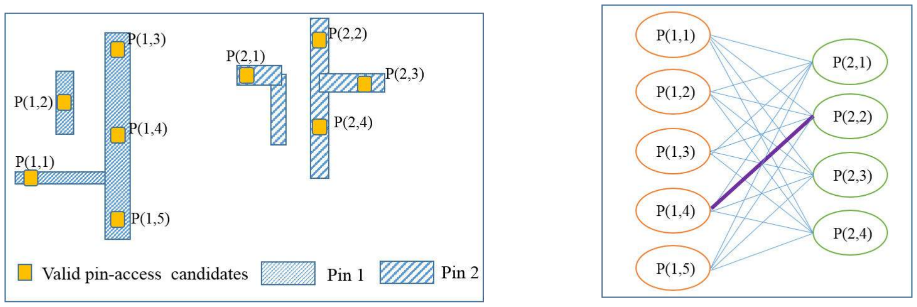
$$Weight = TotalTrackDistance \times 1 + ViaNumber \times 5. \tag{1}$$

With the above graph $G(V, E)$, we choose a root node $x \in V$ (the leftmost point) and generate a minimum spanning tree.

3.2.2 Division-aware components points establishment

Since our net's connecting order are derived from the generated spanning tree, the following step is to effectively extract connection points of every pin of a net. A bipartite graph is used to select net's components points. Each pin shape has one or more valid pin-access candidates, and since too close pin-access points may have space violation, retaining access points for other pins as more as possible is a better choice. So we construct the grid graph as a bipartite graph according to the generated spanning tree. Vertexes in the bipartite graph are two disjoint sets, and weighted edges represent the cost of connecting these two points. We select an edge with the minimum weight in the bipartite graph.

Figure 5 (a) shows that our net component P_1 has five valid pin-access candidates, and another net component P_2 has four valid pin-access candidates. So 20 possible connections and associated weights are generated in this bipartite graph. The selected edge is labeled purple in Figure 5 (b).



(a) (b)

Figure 5: An example to establish net's components connection points

3.3 Optimizing initial routing result

In this subsection, our process is split into two stages to achieve our detailed routing results.

3.3.1 Rectangle-driven initial routing solution

For every edge established in Section 3.2.2, we use A* algorithm to build an initial routing scheme, in which the cost function is the sum of the distance between the current search point and start point and the distance between the current search point and target point. In this routing scheme for an edge, the routing preference metrics may be bad. To improve the metrics, we use the following method to adapt the initial routing.

First, we set up a conflict graph $G = (V, E)$. All initial routing segments of nets on the same metal layer are regarded as vertexes, and each edge $e_{(x,y)}^{(x',y')}$ indicates that two segments $V_{x,y}$ and $V_{x',y'}$ on the same metal layer overlap with each other. Let the weight $W_{x,y}$ of vertex $V_{x,y}$ be the sum of track length without wrong-way routing and off-track routing. To achieve segments with better routing preference metrics, we solve the following problem by using Branch-and-Bound solver to select segments with respect to vertexes of the conflict graph:

$$\begin{aligned} \max \quad & \sum w_{(x,y)} \times c_{(x,y)} \\ \text{such that} \quad & c_{(x,y)} + c_{(x',y')} \leq 1, \quad \text{for any } e_{(x,y)}^{(x',y')} \in E \\ & c_{(x,y)} \in \{0,1\}, \quad \text{for any } V_{x,y} \in V. \end{aligned} \tag{2}$$

In the above formulation, $c_{(x,y)}$ is a binary indicator of deciding whether $V_{x,y}$ is used in the following rerouting stage. The aim of rerouting is to achieve routing segments with improved routing preference metrics. We iteratively rip-up segments with respect to vertexes until there are no conflict edges in the graph.

3.3.2 Initial detailed routing solution optimization

After accomplishing detailed routing with respect to routing preference metrics, we further optimize the detailed routing solution honoring the global routing results as much as possible, and reroute segments overlapped or violating connection constraints. In this stage, our initial detailed routing results are divided into routing segments R_i by via locations. This step can efficiently simplify our method to deal with modern advanced technology nodes.

First, we find open nets and short areas according to modern connection constraints on our track grids. Then for each routing layer, we define a panel L_i to be the i-th metal layer, and R_i to be the set of all routing segments on the i-th metal layer. Let S_{Cost} be the overlapped wirelength of a segment, and O_{Cost} be a binary variable to indicate whether or not this segment is open.

We preprocess all nets according to spacing rules described in modern advanced technology nodes before our detailed routing process [23, 26]. For any via, routing segment and other object of routed nets, we label the rectangle centered at the via, routing segment or other object with width of the minimum spacing as a blockage during the next routing process. This step can satisfy the minimum distance constraint between any two of vias, other objects and segments for the next routing process. By the way, C_{Cost} is set as a binary indicator, which indicates if a segment satisfies the spacing constraint or not.

Minimum area rule specifies that all routing segments in our results should be larger than minimum manufacturable size. In order to satisfy this constraint, we need to insert patch for those segments violating the minimum area rule. Let M_{Cost} be a binary variable indicating whether or not to add patch to a segment.

Due to the particularity of our global routing results and complicated pin information, we need to consider out-of-rectangle wirelength (L_{oc}), wrong wires (L_{wl}), out-of-rectangle vias (V_{oc}) and off-track vias (V_{ot}) metrics in our detailed routing process. Compared with previous connection constraints, these metrics have less influence on our detailed routing results, but are still considered in our cost for optimization.

All the above metrics need to be optimized together for every panel. Therefore, a weighted sum of these costs is generated below to assess our routing results. In equation (3), the coefficients of O_{Cost}, S_{Cost} and C_{Cost} reveal that these metrics have a higher priority to optimize:

$$\begin{aligned} HistoryCost = {} & O_{Cost} \times 100 + S_{Cost} \times 100 + C_{Cost} \times 100 + M_{Cost} \times 1 \\ & + L_{oc} \times 0.5 + L_{wl} \times 0.5 + V_{oc} \times 0.5 + V_{ot} \times 0.5. \end{aligned} \tag{3}$$

In Algorithm 2, we detail the process of reducing history cost of one segment in a panel. It must be noted that, our previous history costs of segments in one panel need to be updated after each iteration. This step will be continued until no segments have any violation of advanced technology nodes.

Algorithm 2: Overall Reduction of Violation of Constraints for a Panel

```
Require: Panel $L_i$, routing segments $R_i$
 1:  for the $j$-th routing segment $R_{ij} \in R_i$
 2:      calculate History Cost of $R_{ij}$;
 3:      repeat
 4:          $H_{max} = selectMaxCostsegmentR_{ij}$;
 5:          $rip-up\ H_{max}$;
 6:          $reroute\ segment\ R_{ij}$;
 7:          $update\ History\ Cost\ of\ R_{ij}$;
 8:      until termination condition is satisfied;
 9:  end for
10:  return $R_i$
```

4 Empirical Studies

In this section, we implement our detailed routing algorithm in the C++ programming language, and test it on modern real industrial benchmarks. All features of each modern industrial benchmark are described in Table 1.

Table 1: Characteristics of modern industrial benchmarks

Benchmark	blk	net	pin	layer	area(μm^2)
case1	0	225	450	6	31 800
case2	0	548	1 149	6	35 130
case3	0	1 862	3 904	6	39 331
case4	4	359	718	9	465 739
case5	4	585	1 377	9	499 887
case6	4	1 534	3 309	9	640 334

There are two kinds of benchmarks. The first kind contains three industrial benchmarks without blockages and vast majority of nets are with two terminals. The second kind of industrial benchmarks is with blockages, the range of routing area, routing layers numbers, and there are a large number of multi-terminals nets. All benchmarks are derived from a single-core 32-bit processor with four memory cores with modern manufacturing design rules. Table 1 presents the exhaustive statistics information of our modern industrial benchmarks, where "*blk*", "*net*", "*pin*", "*layer*", "*area*" give the numbers of blockages and nets, the amount of nets' pins, the range of routing layers, and the die area for each industrial case respectively.

We compare our experimental results generated by our proposed detailed routing process with and without considering the tree-based net's components points selection firstly. And then, we compare our final optimized results (denoted by Ours) with those generated by replacing A* algorithm in our algorithm with the classic depth-first search algorithm (denoted by DFS). All experiments are performed on the same platform with Intel Core 3.40GHz CPU and 16GB memory for fair comparison.

4.1 Effectiveness of tree-based net's components points selection

In Section 3.2, two stages of optimization methods are presented to select our connection points of each pin. In order to reflect the effectiveness of the two stages, we test our algorithm with and without considering the tree-based net's components points selection on the modern industrial benchmarks. The test results are listed in Table 2. In the table, the first column is the names of our industrial benchmarks. The following three columns show the total wirelength ($L_{total}(mm)$), the total numbers of vias (V_{total}) and routing layers (R_l), respectively. The experimental results show that, our algorithm with the tree-based net's components points selection can decrease the total wirelength by 82% averagely, and the number of total vias by 10% on the industrial benchmarks.

4.2 Effectiveness of optimizing initial routing result

In this subsection, we compare our final optimized results (denoted by Ours) with those generated by replacing A* algorithm in our algorithm with the classic depth-first search algorithm (denoted by DFS). The test results are put in Table 3. In the table, five evaluation metrics are listed in the second row, that is, out-of-

Table 2: Comparison of several metrics without and with our tree-based net's components points selection

Benchmark	$L_{total}\times(\mu m)$		V_{total}		R_l	
	w/o	w/	w/o	w/	w/o	w/
case1	802	741	890	885	4	4
case2	2 087	1 952	2 230	2 209	5	5
case3	335 324	8 559	10 546	7 446	6	6
case4	35 658	35 595	2 299	2 318	9	9
case5	129 768	129 584	4 685	4 195	9	9
case6	277 502	276 730	8 578	8 898	9	8
Average	1.828	1.000	1.085	1.000	1.000	1.000

Table 3: Comparison of final detailed routing results between DFS and Ours

Benchmark	DFS results					Ours				
	(L_{oc}) (μm)	(V_{oc})	(L_{wl}) (μm)	(V_{ot})	times (s)	(L_{oc}) (μm)	(V_{oc})	(L_{wl}) (μm)	(V_{ot})	times (s)
case1	1 032.53	237	0	0	3 759.63	632	289	0	0	8.45
case2	2 493.49	603	0	0	8 459.26	256.18	691	0	0	24.12
case3	-	-	-	-	-	667.06	2 318	0	0	69.85
case4	5 719.84	595	0	0	2 719.16	274	664	0	0	448.08
case5	-	-	-	-	-	33 432.38	947	0	0	1 589.62
case6	-	-	-	-	-	72 101.40	2 338	0	0	10 541.60

rectangle wirelength (L_{oc}), out-of-rectangle vias (V_{oc}), wrong wires (L_{wl}), off-track vias (V_{ot}) and actual running time. The experimental results demonstrated in Table 3 show that, our detailed router can complete 100% routability for all different sizes of nets. What's more, it can satisfy the advanced technology nodes as much as possible, and reduce out-of-rectangle wirelength and vias in a reasonable runtime.

Figure 6 shows our final detailed routing results for Benchmark 3 generated by our algorithm. Figure 6 (a) is the full detailed routing result, and Figure 6 (b) is a partial routing result magnified from Figure 6 (a). Furthermore, blue lines in Figure 6 are routed nets and red rectangles are our pin shapes.

5 Conclusion

In this work, an effective algorithm has been presented to solve the modern detailed routing problem. All nets, blockages and pin information are complicated and irregular. The primary purpose of our algorithm is to minimize the total wirelength and total vias by considering the advanced technology nodes. The first two parts of our algorithm focus mainly on preprocessing and local optimizing our pin-access candidates and other connection points of nets. After that, our algorithm completes detailed routing by using an optimized path search method. Experimental results on industrial benchmarks show that, our proposed algorithm not only achieves 100% routability in a reasonable runtime, but also handles connection constraints and honors global routing results efficiently. With the development of modern manufacturing requirements, the detailed routing process with other constraints also need to be further considered in our algorithm.

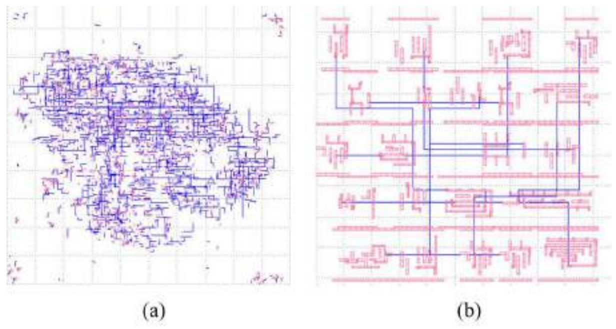


Figure 6: Full and partial detailed routing results of Benchmark 3

References

[1] C.J. Alpert and D.P. Mehta, Handbook of Algorithm for Physical Design Automation, New York: Auerbach Publications, 2008, pp.469-484.

[2] M.M. Ozdal, Detailed-routing algorithms for dense pin clusters in integrated circuits, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **28**:3(2009),340-349.

[3] M.-K. Hsu and Y.-F. Chen and C.-C. Huang and S. Chou and T.-H. Lin and T.-C. Chen and Y.-W. Chang, A novel routability-driven placement algorithm for hierarchical mixed-size circuit designs, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **33**:12(2014),1914-1927.

[4] S.M.M. Goncalves and L.S. da. Rosa. Jr and F.S. Marques, A survey of path search algorithms for VLSI detailed routing, Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), 2017.

[5] C.Y. Lee, An algorithm for path connections and its applications, *IRE transactions on electronic computers*, **10**:3(1961),346-365.

[6] F. Rubin, The lee path connection algorithm, *IEEE Transactions on Computers*, **C-23**:9(1974),907-914.

[7] P.E. Hart and N.J. Nilsson and B. Raphael, A formal basis for the heuristic determination of minimum cost paths, *IEEE transactions on Systems Science and Cybernetics*, **4**:2(1968),100-107.

[8] J. Soukup, Fast Maze Router, Proceedings of the 15th Design Automation Conference on Computer-aided Design, 1978.

[9] A. Heztel, A sequential detailed router for huge grid graphs, Proceedings of Design, Automation and Test in Europe, 1998.

[10] S. Peyer and D. Rautenbach and J. Vygen, A generalization of Dijkstras shortest path algorithm with applications to VLSI routing, *Journal of Discrete Algorithms*, **7**:4(2009),377-390.

[11] B.K. Nielsen and P. Winter and M. Zachariasen, An exact algorithm for the uniformly-oriented steiner tree problem, Proceedings of European Symposium on Algorithms, 2002.

[12] E. Shragowitz and S. Keel, A global router based on a multicommodity flow model, *Integration: The VLSI Journal*, **5**:1(1987),3-16.

[13] P. Raghavan and C.D. Thompson, Multiterminal global routing: a deterministic approximation scheme, *Algorithmica*, **6**(1991),73-82.

[14] C. Albrecht, Provably good global routing by a new approximation algorithm for multicommodity flow, Proceedings of ACM International Symposium on Physical Design, 2000.

[15] J. Seo and J. Jung and S. Kim and Y. Shin, Pin accessibility-driven cell layout redesign and placement optimization, Proceedings of ACM/EDAC/IEEE Design Automation Conference, June, 2017.

[16] X. Xu and B. Yu and J.-R. Gao and C.-L. Hsu and D.Z. Pan, Pin-access planning and regular routing for self-aligned double patterning, *ACM Transactions on Design Automation of Electronic Systems*, **21**:3(2016),1-21.

[17] Z. Heng and Chris Chu, An efficient detailed router with regular routing patterns, *IEEE Transcations on Very Large Scale Intergration(VLSI) Systems*, **21**:9(2013),1655-1668.

[18] X. Xu and B. Cline and G. Yeric and B. Yu and D.Z. Pan, Self-aligned double patterning aware pin access and standard cell layout co-optimization, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **34**:5(2015),699-712.

[19] N. Tim, Gridless pin access in detailed routing, Proceedings of the ACM/EDAC/IEEE Design Automation Conference on Computer-aided design, June, 2011.

[20] Y. Ding and C. Chu and W.-K. Mak, Self-Aligned Double Patterning Lithography Aware Detailed Routing with Color Preassignment, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **36**:8(2017),1381-1394.

[21] J.R. Gao and D.Z. Pan, Flexible self-aligned double patterning aware detailed routing with prescribed layout planning, Proceedings of the ACM international symposium on International Symposium on Physical Design, 2012, pp.25-32.

[22] F.O. Hadlock, A shortest path algorithm for grid graphs, *Networks*, **7**:4(1977),323-334.

[23] Y.-H. Su and Y.-W. Chang, Nanowire-aware routing considering high cut mask complexity, *IEEE Transactions on Computer-Aid Design of Integrated Circuits and Systems*, **36**:6(2017),964-977.

[24] J.Y. Hsiao and C.Y. Tang and R.S. Chang, An efficient algorithm for finding a maximum weight 2-independent set on interval graphs, *Information Processing Letters*, **43**:5(1992),229-235.

[25] C.C. Huang and H.-Y. Lee and B.-Q. Lin and S.-W. Yang and C.-H. Chang and S.-T. Chen and Y.-W. Chang and T.-C. Chen and I. Bustany, A detailed-routing-driven placer for mixed-size circuit designs with technology and region constraints, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **37**:3(2018),669-681.

[26] I.-J. Liu and S.-Y. Fang and Y.-W. Chang, Overlay-aware detailed routing for self-aligned double patterning lithography using the cut process, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **35**:9(2016),1519-1531.

[27] S.K. Han and K. Jeong and A.B. Kahng, Stability and scalability in global routing, Proceedings of IEEE International Workshop on System Level Interconnect Prediction, June, 2011.

(*edited by Mengxin He*)